Computer Communications 85 (2016) 58–73

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Efficient message delivery models for XML-based publish/subscribe systems

Yang Cao[a], Chung-Horng Lung[b,*], Shikharesh Majumdar[b]

[a] School of Computer Science, Carleton University, Ottawa, Ontario K1S5B6, Canada
[b] Department of Systems and Computer Engineering, Carleton University, Ottawa, Ontario K1S5B6, Canada

ARTICLE INFO

Article history:
Received 24 June 2015
Revised 6 March 2016
Accepted 30 March 2016
Available online 5 April 2016

Keywords:
XML publish/subscribe system
Network services and applications
Application layer multicast
Peer model
Performance evaluation

ABSTRACT

XML-based publish/subscribe (pub/sub) systems have been receiving a great deal of attention from the academic community and the industry. This research focuses on efficient pub/sub systems and considers the devising of XML-based pub/sub systems from the perspective of subscription/query and publication message delivery. Existing research mainly focuses on the efficiency of XML publication message filtering algorithms. Not much research, however, has considered using the system or the communication model in the context of XML publication messages delivery. This paper presents innovative XML delivery techniques, the cross-layer model and the peer model; both techniques make use of publisher and customer edge brokers for efficient XML subscription aggregation and publication message delivery. The primary contribution of the proposed models is the reduction of the number of XML publication message filtering and XPath query aggregation operations performed in the conventional filter-based XML multicast model, which has a high computational overhead. The main idea is to store user subscriptions at customer and publisher edge brokers which are either directly connected or close to the subscribers and the publisher, respectively. We have performed a number of experiments within a controlled local area network (LAN) environment for demonstration of the basic concepts underlying the techniques and in the Amazon cloud environment that emulates the wide area network (WAN). Both the cross-layer and the peer models can reduce the end-to-end (E2E) delay in message delivery. For example, the results obtained from experiments in a LAN demonstrate several-fold performance improvement in E2E delay for both the cross-layer and the peer models compared to the conventional filter-based XML multicast model, and the results using the cloud show an improvement as high as 64% in E2E delay for the peer model over the multicast model.

1. Introduction

As more information becomes available on the Internet, both information users and producers face more challenges, stemming from both the volume of information and the format variety of information such as HTML, XML and JSON. XML has been used for publish/subscribe (pub/sub) systems for message delivery. In pub/sub systems, subscriptions or queries are submitted by users, which are forwarded to a publisher, whereas messages are sent by the publisher to interested subscribers. In XML (Extensible Markup Language, [9] pub/sub systems, messages from the publisher are encoded in XML and user subscriptions are specified using the XPath syntax [28,68].

XML has received a great deal of attention in the research community [2]. XML is a markup language for creating documents containing semi-structured information. Various features are available in XML and XPath expressions. Our paper currently supports the key XPath features, including the parent/child operator ('/'), ancestor/descendant operator ('//'), wildcard ('*') and predicate ('[]'), i.e., $XP^{/,//,*,[]}$. In this paper, the following pairs of terms are used interchangeably: user and subscriber, (user) subscription and query, (publication) message and document, and node and broker.

A number of techniques have been proposed to improve XML processing, including XML pattern matching [47,48], XML keyword searching [65], efficient XML queries evaluation [40,46,67], data structures or labeling schemes for XML tree management ([33,43,45,70]), parallel processing of XML queries matching for performance improvement [51,52]; and message dissemination and delivery (see the following paragraphs for more detailed description). This paper focuses on XML message dissemination and delivery for pub/sub systems. Specifically, the paper

* Corresponding author. Tel.: +16135202600.
E-mail address: chlung@sce.carleton.ca (C.-H. Lung).

http://dx.doi.org/10.1016/j.comcom.2016.03.023
0140-3664/

deals with the development of efficient application-layer delivery mechanisms—the *cross-layer model* and the *peer model*—and performance evaluation of these models for XML message dissemination and delivery from the publisher to interested subscribers.

XML message dissemination and delivery. XML message delivery or routing in a pub/sub system include two main operations: (i) XPath query summarization/aggregation, and (ii) XML message filtering/matching to identify matched subscribers. To match a query that contains a combination of XPath operators against a publication message, the computational cost for the matching operation can be high for complex and nested queries. The performance of XML message delivery or routing plays a crucial role for XML based pub/sub systems, as the volume, variety, and velocity of information have been increasing dramatically.

Most existing XML pub/sub systems are based on the XML multicast model. With this model, the aforementioned two operations are performed at application-layer XML-capable nodes or brokers based on a multicast tree, rather than regular network-layer routers. Moreover, most existing approaches to XML routing emphasize on the efficiency of those two operations from the data structure or algorithmic perspective. A representative approach is Yfilter [31]. See Section 2 for more related filtering techniques. However, based on the multicast model, those two operations are performed at each intermediate XML broker along the multicast tree, which can significantly affect the overall system performance and scalability.

On the other hand, performance improvement can also be tackled from the communications system or communication model aspect as well. One of the most successful examples using communications system for performance improvement is content delivery networks (CDNs) [37]. The concept of CDNs is straightforward, but the benefits of CDNs are enormous for bandwidth usage savings and delay reduction. Other highly successful communication models for efficient management and cost reduction include the MPLS-based peer-to-peer model [36] that has been widely adopted for virtual private networks (VPNs) and the recent popular software-defined networking (SDN) [42].

The motivation of this research is to investigate communication models for performance enhancement for XML pub/sub systems. Specifically, the main objective is to reduce or remove the aforementioned expensive XML query aggregation and message filtering operations from the intermediate XML brokers. The paper is an extension of our initial paper using the cross-layer model [14]. (Note that we have renamed the approach as cross-layer model in this paper.) In comparison to our earlier paper, a new peer model is introduced in this paper, which is also compared with the conventional XML multicast model and our previous cross-layer model. A short description of the key features of each of related models XML pub/sub message delivery is presented as follows. Section 3 presents a more detailed description for different models.

- *Unicast model.* This model is based on the traditional unicast approach that does not use query aggregation or XML publication message filtering. A unicast is established from the publisher to each matched subscriber for the publication message delivery. The main problem with the unicast model is that the number of messages can be very large and some messages will also be duplicated. Subsequently, the delay and network bandwidth usage can be large.
- *Application layer XML multicast model.* Generally speaking, multicast can be supported at the network layer or the application layer. Multicast has become popular, mostly for multimedia applications [49]. The primary advantage of multicast is the increased efficiency resulting for the reduction of traffic redundancy. Nevertheless, there are limitations or disadvantages with the popular network layer multicast for streaming applications due to its use of User Datagram Protocol (UDP). Although UDP is suitable for multimedia applications, it may not be desirable for applications that require reliable transmissions, as packet drops, duplicates and out of order delivery are to be expected for UDP [66]. In addition, providing multicast services increases a great deal of complexity on Internet service providers [58], such as the state information for each multicast distribution tree in the network needs to be provided and maintained.

The overlay model has been proposed for application layer multicast services with specialized brokers and a broker topology built above the network layer. The application layer overlay model offers additional computational power, as existing network devices are unsuitable for the application layer multicast due to limited resources and high-performance demands for routing and forwarding tasks. The application layer multicast model has been adopted for many XML filtering and matching approaches. Examples include Yfilter [31], Afilter [10], Gfilter [26], Bfilter [29], Efilter [38], Pfilter [61], and Wfilter [50]. Section 3 describes Yfilter in more detail, as it is the most popular XML multicast model. In this model, a logical application layer multicast network is constructed from a publisher to subscribers. The main problem with these methods is their high cost for subscription (or query) aggregation and XML publication message filtering operations that need to be performed at each XML-capable broker at the application layer.

- *Cross-layer model.* The cross-layer model was proposed in [14]. The proposed cross-layer model removes the XML query aggregation and publication message filtering operations from intermediate brokers. Instead, these two operations are only performed at edge brokers that include both publisher edge (PE) and customer edge (CE) brokers. Each PE is connected to the publisher and each CE is connected to a number of subscribers. However, the model still creates and maintains a similar application layer message delivery tree used in the multicast model. Each CE forwards the queries it has received from the subscribers to a PE through XML-capable intermediate brokers and the PE will perform the query aggregation operation for queries received from all CEs to build an aggregated query tree. When a publisher sends a publication message to a PE, the PE will perform a filtering operation to identify the matched CEs and forward the message to those CEs through intermediate brokers.

This paper extends the investigation of the cross-layer model further from a LAN setting used in [14] to a WAN environment using multiple Amazon data centers.

- *Peer model.* This paper proposes a new peer model for the XML pub/sub systems. The proposed peer model adopts the same concept of using edge brokers as that used in the cross-layer model for query aggregation and publication message filtering operations. Hence, PE(s) and CE(s) perform the same operations as that in the cross-layer model. However, the peer model differs from the cross-layer model that builds and maintains the application layer multicast tree or delivery paths. In the peer model, a multicast tree is no longer needed. Specifically, a PE has a direct logical (not physical) connection to CEs. When the publisher sends a publication message to a PE, the PE performs a filtering operation to identify the matched CEs and forwards the publication message to the matched CEs using the regular network layer IP routing/forwarding technologies, instead of the application layer multicast forwarding.

Therefore, the cost, i.e., both capital expenditure (CAPEX) and operational expenditure (OPEX) can be greatly reduced with the peer model, as no specific arrangement of XML-capable intermediate routers across a geographically distributed area is needed,

and the cost for a multicast tree establishment and maintenance is eliminated. We have performed extensive experiments using Amazon datacenters around the world to emulate the WAN to investigate the peer model and compare the efficiency of these three models. Section 5.4 presents the experimental results.

Section 3 presents a detailed description for all those models and the popular Yfilter. Yfilter is used as the filtering technique for comparing different models in this paper. Note that query aggregation is a separate topic, as it involves complex algorithms for XPath queries. Various query aggregation techniques and related labeling algorithms have been discussed in [13,15,22,24,33,35,43–45,69]. Those algorithms can be used in any of the aforementioned multicast, cross-layer, or peer model. Hence, this paper assumes that queries have been aggregated and only focuses on XML publication message filtering and forwarding.

The rest of the paper is organized as follows. Section 2 highlights related work on various application layer XML routing approaches. Section 3 explains in details the three models under investigation—the traditional XML multicast model, the cross-layer model, and the peer model. Section 4 explains how experiments are set up. Section 5 presents the performance evaluation results. Section 6 concludes the paper.

2. Related work

This section addresses existing application layer XML multicast routing technologies for a pub/sub system, which can be grouped into four classes: conventional filter-based XML multicast, channelization XML message routing, rendezvous-based XML message routing, and other models. For an unstructured overlay topology, conventional filter-based XML message routing and channelization XML message routing can be used for XML message delivery. The rendezvous-based XML message delivery model is proposed for a structured overlay topology. Other XML routing models are proposed in specific contexts, e.g., a VPN environment or an integrated existing pub/sub system. Each of the models is described in the following sections.

2.1. Conventional filter-based XML multicast

With filter-based message routing, all user subscriptions or queries are stored and aggregated at each node along the path from the leaf node to the root node of a distribution tree rooted at a publisher. In turn, a published message needs to be repeatedly evaluated at each intermediate node or broker along the paths in the tree. A message is forwarded to a node if matched subscription(s) of the message can be found.

A number of filtering techniques for XML pub/sub systems have been proposed in the literature. Xfilter [3] is one of the first XML filtering techniques based on deterministic finite automata (DFA). It stores user requests and handles each request individually, which may be redundant for similar queries. Xfilter can handle more complex XPath relationship, such as ancestor/descendant (represented by a '//' in XPath) as well as a wildcard '*', than simply the linear parent/child relationship.

Yfilter [30], instead, adopts nondeterministic finite automata (NFA) to deal with the state explosion problem that DFA has. YFilter emphasizes prefix sharing and builds a single combined NFA, which significantly improves system performance. However, the ancestor/descendant relationship in complex queries may result in the number of active states increasing exponentially [50]. Post-processing is another extra task for Yfilter to deal with queries with nested paths. YFilter decomposes a complex query into simple ones and matches them separately. For example, a complex query */a[/d]/b/c* is decomposed into two simple queries: */a/d* and */a/b/c*. Consequently, the processing time may be high for complex queries.

ONYX [32] is one of the first XML-based pub/sub systems using the filter-based XML message routing scheme. ONYX uses Yfilter [30] as a filtering engine. XNET is another representative filter-based XML message routing system [22–24,21]. Xtrie [20] is the filtering engine in XNET. XNET supports reliability in case of network failures and manages dynamic subscriptions. In [44], the authors present an XML-based pub/sub system using filtering-based XML message routing. Although the publisher-based distribution tree is optimal in ONYX and XNET in terms of the number of forwarded messages, it is expensive to maintain when the number of publishers is large [34].

On the other hand, Afilter [10] uses both prefix and suffix commonalities (as opposed to only prefix commonalities adopted in Yfilter) to reduce mismatches. Afilter organizes user queries or requests as a directed graph, and uses stacks and a triggering mechanism to delay the message matching task until a trigger condition is met. Afilter was found to be more efficient than Yfilter [10], but Afiter does not consider predicates for queries.

Gfilter [26] improves the post-processing of complex queries using the Tree-of-Path coding scheme. Gfilter focuses on improving the path matching via a bottom–up approach instead of a top–down mechanism, which results in faster filtering decision. However, decomposition and post-processing are needed for handling complex queries, which increases the overall processing time.

Hfilter [64] combines lazy DFA and NFA techniques, because NFA based approaches are more space efficient than lazy DFA based techniques, but lazy DFA based techniques could be more time efficient than NFA based methods. However, the large space consumption problem still exists, resulting in memory overflow and degradation of efficiency [64].

Bfilter [29] uses lowest-level branch points as starting points for message processing, as opposed to a top–down (e.g., YFilter) or bottom–up (e.g., Afilter and Gfilter) approach. The filtering process will recursively utilizes higher-level branch points for triggering matching operation. The main idea is based on heuristic that the probability of mismatching that will occur at the branch point is expected to be the lowest in comparison to top–down and bottom–up approaches. The probability of mismatching is lower means that the filtering process can be completed faster, because mismatching in the XML message filtering process causes the filter to spend time on evaluating queries that will ultimately fail.

Wfilter [50] utilizes a coding scheme to convert each query into a special internal representation. The approach can efficiently store and identify query nodes for the message matching process. However, the approach only focuses on linear parent–child and ancestor–descendant relationships. Nested paths and queries with branches are currently not considered.

Pfilter [61] is a sequence-based filtering technique. Pfilter encodes queries into value-based sequences using pre-order traversing of the query tree. The objective is to identify related profiles for structure and content matching. The generated sequences are grouped into hash-based indices that can be processed concurrently.

Efilter [38] incorporates the advantages of FiST [41] and Sfilter [1]. FiST is another sequence-based method that converts queries into sequences which are organized as tree structures. The filtering task is to find matched sequences. The matching is efficient using FiST, but it is restricted, because the nodes in a twig pattern need to follow the document order in XML. Sfilter utilizes a query guide and simple integer stacks (as opposed to a state stack used in YFilter) to improve efficiency. A query guide represents all the path trees for the queries. However, Sfilter focuses on linear paths; it does not handle predicates, twig queries (queries with twig

patterns or branches), or nested paths, which are important features for XML documents.

All those aforementioned methods focus on the data structure or algorithmic aspect for XML message filtering. Although those approaches use different filtering techniques, they share a common communication model: each broker needs to perform the filtering operation and find the matched subscriptions for XML message delivery. The publication message will then be forwarded to the next broker on the multicast tree. As stated in Section 1, the main problem with the conventional filter-based XML multicast model is the computational overhead that is needed for each broker. Section 3 describes in detail the conventional XML filter-based multicast model and presents a comparison of this model with the other two XML message delivery models.

The authors in [11] and [12] propose a new architecture for content-based pub/sub network, called Match Early with Dynamic Multicast (MEDYM). The MEDYM system matches subscriptions from remote servers to calculate a destination list of servers with matching subscription, then routes a message to the destination servers. A broker computes the next hops and a new destination list for each next hop as it receives a message. Similar to our proposed peer model, the complex matching operation is also only performed at network edge using the approach. But, unlike our proposed peer model, the intermediate brokers in MEDYM need to compute an approximate minimum spanning tree across destinations associated with the matched messages for multicast routing using a shortest-path-MST routing algorithm.

2.2. Channelization XML message routing

XTreeNet is an architecture for supporting XML-based systems, which integrates the pub/sub and the query/response model [34]. Published messages and user subscriptions are mapped to sets of *content descriptor* (CD). A published message matches a subscription when their associated CDs have at least one common CD. A distribution tree is established for each CD that is rooted at each publisher. Only the first hop (source overlay) broker needs to map messages to CDs, whereas the intermediate brokers only need to do forwarding. XTreeNet uses CDs as channels to alleviate repeated matching of published messages against subscriptions at multiple nodes in an overlay network. But XTreeNet does not support XPath query aggregation. In the SemCast [54] system, there are source brokers, gateway brokers, and internal brokers. The source brokers connect to publishers and the gateway brokers connect to subscribers. The coordinator is responsible for channelization management. It communicates with the source brokers and gateway brokers. SemCast aggregates and merges subscriptions into channels. A published message is matched to such aggregates. After matching, each published message needs to be mapped to corresponding channels and forwarded to a specific channel(s). Each channel is determined in a centralized manner using a cost-based model and is implemented as a multicast tree. Subscriptions are sent to the coordinator to check whether an existing channel covers it using a containment algorithm, then join corresponding trees. Our proposed peer model uses the IP network layer for message delivery from PE to CEs and there is no need to maintain CDs and the distribution tree for each CD, as used in [34], or to have a central coordinator to manage channels and the multicast tree. XTreeNet and SemCast cluster messages and queries to semantic topics and remove content-based XML message filtering. Our peer model supports content-based XML message matching.

2.3. Rendezvous-based XML message routing

Net-X [59] is a proposed pub/sub system built on top of a distributed hash table. Net-X uses signatures to discover interest match between user interests and published document. The published document can be either small text files or MPEG7 movies, which requires a different distribution tree for different documents. Net-X builds a per-document tree. For each published document, there is an overlay distribution tree rooted at the publisher only connecting the users who are interested in the specific document. In Net-X, XPath subscriptions and published XML documents are mapped to polynomial signatures and are stored in a distributed hash table (DHT) on which subscriptions and published documents can be matched.

Hermes [57] is an example of type-based pub/sub systems, which has an XML-based API. Hermes uses a network of event brokers built on top of a structured overlay network for rendezvous-based routing and fault-tolerance.

To provide an efficient network delivery, Net-X and Hermes are implemented via a structured overlay-DHT, while our peer model is implemented via an unstructured overlay.

2.4. Other models

In [8], the author proposes to deliver a pub/sub XML system within a VPN. A VPN is implemented on top of a backbone network, such as an IP/MPLS backbone [8]. Each VPN has a unique identifier. An XML router can be embedded in a publisher edge (PE) node or connected to a PE. An XML router can communicate on multiple carrier-based VPNs crossing a PE based on unique identifiers [8] and deliver the publication messages within the same VPN.

In [56], the authors propose a common API for pub/sub systems in order to build an extensible system on top of existing pub/sub systems. XML messages are defined using the XML-RPC data model and subscriptions are described using a lightweight XPath grammar. The API supports three levels of compliance. Level 1 compliance specifies abstract operations; level 2 compliance describes interactions using an XML-RPC mechanism; and level 3 compliance wraps different pub/sub systems and hides the semantic differences of events and subscriptions used in the systems. Siena [16–18], Hermes [57], and Scribe [19] are used as case studies in [56].

Our proposed peer model is not limited to VPN environments and the VPN service provided by ISPs, as advocated in [8]. In addition, our proposed peer model does not need to construct and maintain a multicast tree. Our peer model focuses on efficient query and message delivery for XML pub/sub systems.

The paper [6] presents a complementary approach to the conventional application layer multicast by reorganizing the overlay network topology based on the similarity of interests. For the conventional application layer multicast, some intermediate nodes (pure message forwarders) act as event routers, even though they have no subscription matching event to be transmitted. The use of pure forwarder type of nodes is considered inefficient. The self-organization algorithm proposed in [6] includes: triggering of a broker overlay, broker discovery, tear-down link selection and overlay topology update. However, it is not enough to use only the similarity of interests between brokers, because the brokers sharing high degree of similarity may have a long geographical distance between them. Overlay networks are usually built by considering constraints based on physical proximity between brokers, such that the resulting overlay topology closely matches the underlying network topology. Moreover, measuring the similarity of interests between brokers and other operations can be expensive for XML messages and XPath subscriptions.

The broker overlay of a pub/sub system can impact system performance. Existing approaches to constructing broker overlay build separate dissemination overlays on a per-topic basis. But the resulting overlays can exhibit high average node degrees [27,25]. The

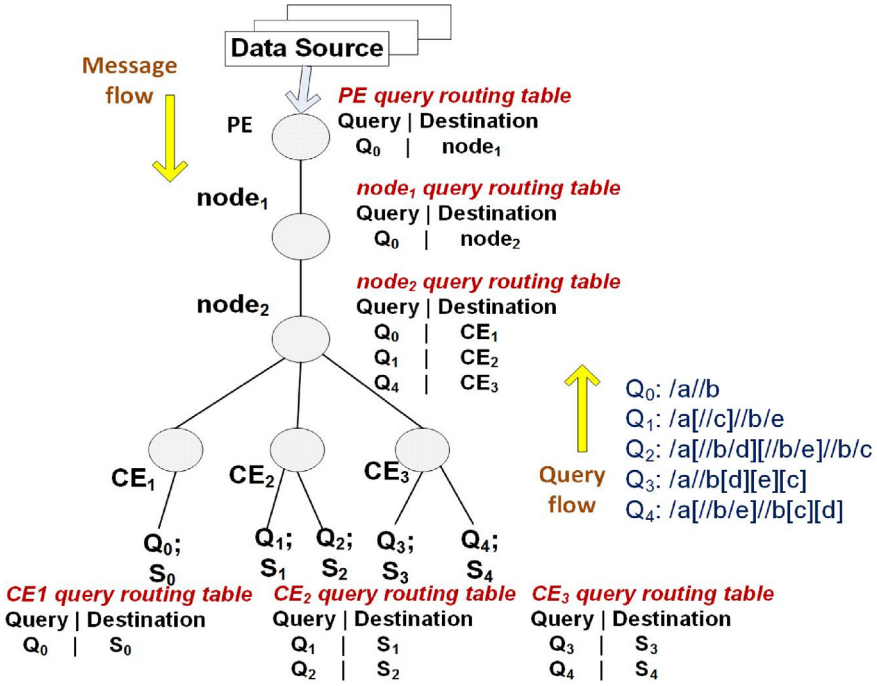

Fig. 1. XML pub/sub with the conventional filter-based XML multicast model.

paper [25] addresses the topic-connected overlay (TCO) construction problem for topic-based pub/sub systems in a homogeneous environment, while the paper [62] discusses the same problem in a heterogeneous environment. The TCO problem is an NP-complete problem [27]. Greedy merge algorithms for solving the set covering problem are proposed to find an approximate solution for the TCO problem. The algorithms presented in [25,62] divide the original TCO problem into several sub-overlay construction problems (referred as sub-TCO problem) by random partitioning. Each sub-TCO problem is solved independently. Solutions to the sub-TCO problems are subsequently combined to one TCO solution for the original TCO problem. The two papers discussed above focus on how to construct high-quality broker overlay topologies, while our work focuses on devising of efficient routing protocols.

2.5. Summary

Among the existing application layer XML pub/sub models, the filter-based multicast approach is the most widely studied. A tremendous amount of research has been devoted to investigating various filtering techniques, as described in Section 2.1. Although some filtering techniques have demonstrated higher performance than others, a common drawback of those filtering-based techniques is that both query aggregation and publication matching operations need to be carried out on each broker along the multicast path. As a result, E2E delay becomes high for the filter-based multicast model. The main contribution of our paper is to reduce the aforementioned computational cost at each broker, and hence E2E delay, by utilizing different communication models, e.g., the cross-layer model and the peer model, which eliminate those two operations at intermediate brokers. Section 3 discusses these two models in detail. We have performed thorough experimentation in a LAN environment to validate the concept and in a WAN environment to measure and compare realistic E2E delay for all three models.

3. XML message delivery models

This section compares the conventional filter-based XML multicast model, the cross-layer model, and the peer model. We first describe the conventional filter-based XML multicast model and explain the core XML publication message filtering operation using the Yfilter [31] technique, as Yfilter is the most popular filtering technique. Other filtering techniques, as described in Section 2.1, also adopt the same model from the network or system perspective. Following that, we present the system architecture for the cross-layer model and the peer model; any other filtering techniques, e.g., Afilter, Gfilter, etc., highlighted in Section 2.1, can also be used together with the new models.

3.1. Conventional filter-based XML multicast model and Yfilter basics

The conventional filter-based XML multicast model, as shown in Fig. 1, has been used by a number of approaches, including Siena [16], Yfilter [31], Afilter [10], Gfilter [26], Sfilter [1] Wfilter [50], Pfilter [61], Efilter [38], and Bfilter [29]. Although those approaches use different data structures or algorithms for performance improvement, they share one common characteristic: *each broker or node* in a pre-established multicast tree conducts both query aggregation and XML publication message filtering operations. XML queries are aggregated to reduce the number of multicast messages that need to be transmitted over the network. However, as highlighted in Section 1, the computational overhead of these two operations can become high with this communication model, as each broker along the multicast path needs to perform these two operations. Moreover, the cost (both CAPEX and OPEX) will also be high for setting up and maintaining the application-layer multicast tree.

As query aggregation and publication message filtering/forwarding are two key operations, they are explained further here. Query aggregation needs to consider if a query is covered by

existing queries, or vice versa. The query covering relationship is defined as follows.

Definition 1. For two XPath queries Q_i and Q_j and an XML message m, a covering relationship (partial order) for Q_i and Q_j holds if every XML message m that matches Q_j also matches Q_i, i.e., Q_i covers Q_j (denoted $Q_j \subseteq Q_i$).

The purpose of Fig. 1 is to illustrate the query routing and forwarding using the conventional filter-based XML multicast model: query Q_0 is present at CE_1; Q_1 and Q_2 are present at CE_2; Q_3 and Q_4 are aggregated at CE_3. CE_1, CE_2, and CE_3 forward the aggregated queries to their upstream broker, $node_2$. In turn, $node_2$ aggregates the queries and builds a query routing table, then forwards the aggregated query tree Q_0 to $node_1$. For this specific example, $node_1$ then forwards Q_0 to PE without query aggregation, as Q_0 is the only query at $node_1$. Finally, PE receives query Q_0 from $node_1$ and builds a query routing table. A PE can also be connected to multiple brokers in other scenarios. The query routing table at each broker may be different depending on the network topology and queries. Later, when a broker receives an XML publication message, the XML filtering engine (e.g., Yfilter) matches the arriving XML messages against user XPath queries. As illustrated in Fig. 1, query aggregation and message filtering operations are performed at each node along the multicast tree. Hence, the overhead for the conventional filter-based XML multicast model at each broker is high, especially when the aggregated tree and the XML publication message are large or complex.

Yfilter [31] is based on nondeterministic finite automaton (NFA) that has been proposed for the conventional filter-based XML multicast model. Yfilter uses a finite state machine (FSM) to represent a subscription, where each location step of a subscription is mapped to a state. A location step has three parts: an axis, a node test, and zero or more predicates. The axis specifies the tree relationship between the context node and the nodes selected by the location step. For example, child (/), parent, ancestor, descendant (//), following-sibling, preceding-sibling, following, preceding, attribute (@), namespace, self (.), descendant-or-self (.//), and ancestor-or-self are typical axes in XPath [28]. The node test specifies the node type and the nodes selected by the location step. The predicates are expressions which can refine the set of nodes selected by the location step. Thereby, XML publication messages matching XPath queries are reduced to matching XML publication messages against an NFA instead of all XPath queries.

Fig. 2 demonstrates an example aggregated query tree at $node_2$, as depicted in Fig. 1, and its NFA representation in Yfilter. The aggregated query tree covers query Q_0, Q_1, and Q_4. Each circle in the Fig. denotes a state, e.g., state 1 to state 7. Two concentric circles denote an accepting state, e.g., state 3 to state 7; such states are associated with the IDs of the queries they represent. A directed edge with a solid arrow represents a transition. The symbol on an edge represents the input that triggers the transition. The special symbol "$*$" matches any element. The symbol "$å$" is used to mark a transition that requires no input. The operator "//" of a subscription is mapped to a combination of an "$å$" transition and a self-loop transition with a "$*$" input tag. The operator "//" results in a state being visited many times. The common prefixes of all the queries are shared in Yfilter. The NFA contains multiple accepting states. While each query in the NFA has only a single accepting state, the NFA represents multiple queries. Identical (and structurally equivalent) queries share the same accepting state.

A Simple API for XML (SAX) parser [60] is commonly used to drive state transitions of the automata to process the filtering task. State transition occurs in case of a state match. Yfilter processes an XML publication message by searching from the root of the NFA. Yfilter tracks all matched transition states for each start tag. There is a selection operator for each accepting state to handle

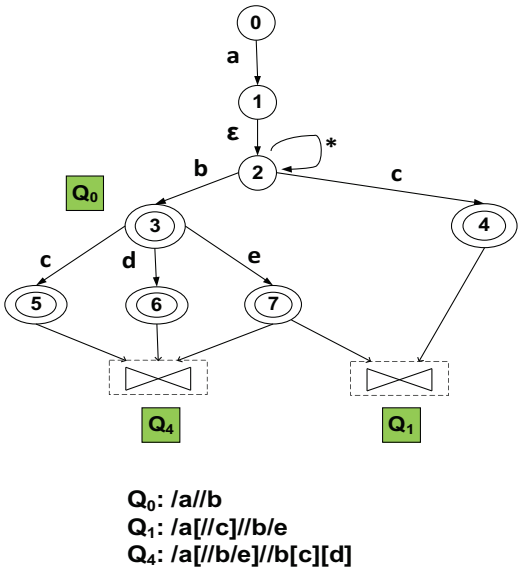

Fig. 2. Example queries and their representation in Yfilter.

the value-based predicate in the corresponding path. For a tree-structure XPath query, Yfilter splits such a query into a set of multiple simple path expressions path by path. Yfilter applies a join operation (∞) after selections. The join operator validates the correctness of the tree structure for an XPath query.

In summary: Yfilter is efficient, as it minimizes the number of publication messages that needs to be forwarded to interested subscribers. But the computational cost for query aggregation and message filtering can be high, especially if complex XML operators are used in queries. Further, these two operations need to be performed at each broker which can increase the E2E delay significantly.

3.2. System architecture for the cross-layer model and the peer model

As explained above, XPath query aggregation and XML publication message filtering are complex operations that are performed with XML-capable brokers. Both the cross-layer model and the peer model eliminate those two complex operations for intermediate brokers. The cross-layer model is shown in Fig. 3(a) and the peer model is shown in Fig. 3(b). The difference between the cross-layer model and the peer model lies in the fact that the cross-layer model still utilizes the same application layer XML-capable intermediate brokers as used in the multicast model for forwarding; whereas for the peer model, no intermediate brokers are needed at the application layer. Put it in another way, the cross-layer model still makes use of the application layer broker topology or the multicast tree for query and message forwarding. But for the peer model, the PE(s) has (have) a direct logical (not physical) connection with all the CEs at the application layer. The idea is similar to the peer-to-peer model in VPNs, where PEs are logically connected. In our proposed peer model, message delivery between the PE and CEs is realized using the network layer routing and forwarding technologies, rather than the overlay application layer XML multicast model. Existing XML filtering techniques, such as Yfilter, Afilter, Efilter, Bfilter, Gfilter, Pfilter, Wfilter, and etc., can still be used together with either the cross-layer model or the peer model at PEs and CEs using these two models.

In Fig. 3(a), queries are aggregated at CEs. Each CE then sends aggregated queries to the PE following a pre-established multicast tree or path. Intermediate brokers, e.g., node 1 and node 2, do not perform further query aggregation as in the case of the multicast

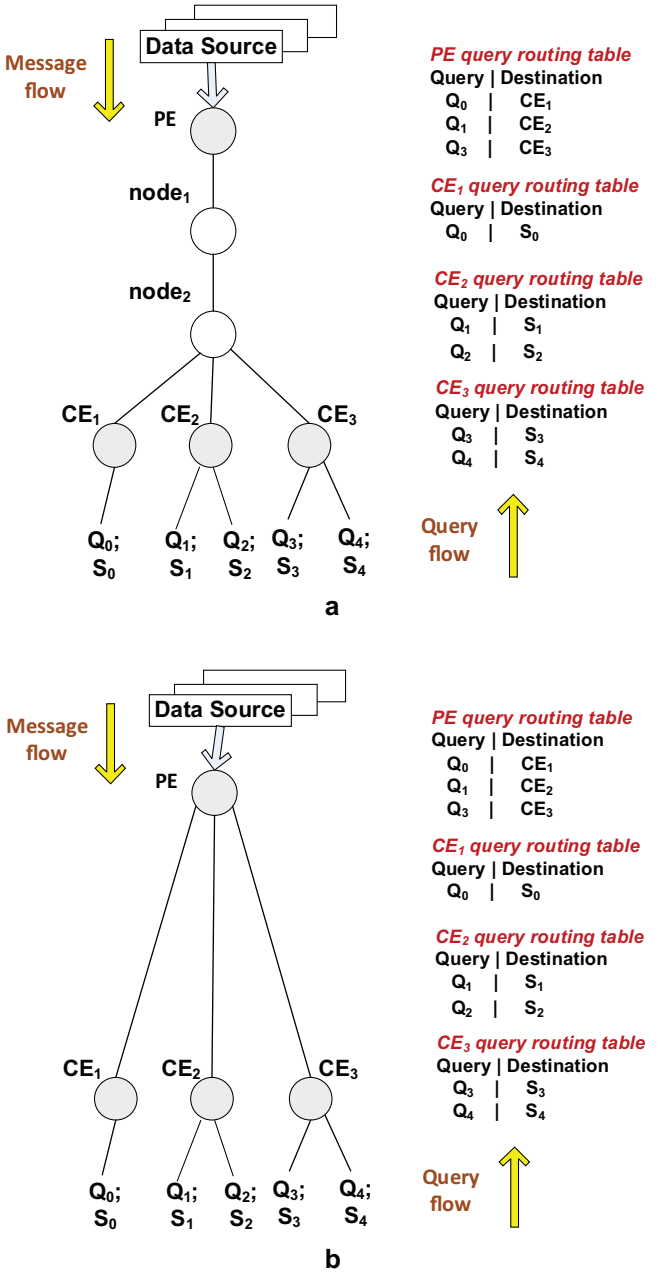

Fig. 3. XML pub/sub: (a) the cross-layer multicast model (queries used are listed in Fig. 1); (b) the peer model.

model. The PE also performs the query aggregation operation after it receives the queries from CEs. When a publisher sends a publication message to a PE, the PE executes the filtering operation to identify the matched CEs. The message is then forwarded to those matched CEs via the paths of the multicast tree. When a CE receives a publication message, it does the filtering operation again to identify the matched subscribers and forwards the publication message to those matched subscribers. Hence, in the cross-layer model, the query aggregation and message filtering operations are only performed at edge brokers, e.g., PEs or CEs, not at intermediate brokers. Intermediate brokers in this case are used for construction and maintenance of the multicast tree for XML-capable brokers.

Similar to the cross-layer model, with the peer model, as illustrated in Fig. 3(b), the two complex operations—query aggregation and message filtering—are also only performed at PEs and CEs. However, the peer model takes one step further by using existing network routing and forwarding technologies for query and message delivery without the need of deploying specialized XML-capable brokers performing multicast forwarding at the application layer, as used in the conventional multicast model and the cross-layer model. With the peer model, after a PE receives a publication message and identifies the matched CEs via the filtering operation, the PE then simply forwards the publication message to those matched CEs using the regular network layer IP routing and forwarding technologies, or simply the Internet technologies. Without those special XML brokers and specific application layer topology, the setup and maintenance can be realized much more efficiently.

Although the work in [8] shares some similarities with our model, the difference is that [8] targets the virtual private network (VPN). Our peer model can be applied in the public Internet without VPN services. With the new peer model, the service provider can install the XML aggregation and filtering features at PEs and CEs to perform the XML message filtering and XPath query aggregation operations, as shown in Fig. 3(b).

3.3. Changes of subscriptions or network topology

In practice, queries can be changed and new queries can be added dynamically, Hence, this section concerns the changes of user subscriptions/queries or the network topology due to failure(s). First, we describe the effect of addition or removal subscriptions or queries using the aforementioned three models. The conventional multicast model produces a minimal query aggregation tree in case of subscription additions or removals, because the aggregation operation is performed at every broker whenever there is a subscription update on the path from a CE to a PE. On the other hand, when there are subscription updates for the cross-layer model, no query aggregation operation is incurred at intermediate brokers. Similarly, for the peer model, there are no intermediate brokers at the application layer.

For both the cross-layer model and the peer model, when a CE receives a new query, the CE performs the aggregation operation and sends the aggregated query to a PE. There are three possibilities: (i) the new query is covered by the existing query tree; (ii) the new query covers existing queries; and (iii) the new query is totally independent of the query tree. In case (i), no operation is needed, except that the new query will be stored in the query table at the CE. For case (ii) the new query will be added to the query table at the CE and the new aggregated tree will be sent to the PE. In case (iii) the new query will be sent to PE and then PE will perform the aggregation operation.

The next scenario is related to changes of network topology due to failure(s). The conventional filter-based XML multicast model is designed at the application layer. It provides multicast functionalities above the IP layer, i.e., a multicast tree is built above the network layer and data are transmitted between neighbors in the multicast tree. In case of a node or a link failure, for the conventional multicast model and the cross-layer model, a dedicated special recovery mechanism is required in order to reconstruct the multicast tree and follow the XML filtering and routing mechanism.

For the peer model, on the other hand, IP network layer recovery mechanisms will be used in case of a node or a link failure, which already are supported by ISPs. In Fig. 4, when there is a link failure for Path 1, the peer model can simply use the IP/MPLS network layer recovery mechanism and re-route the traffic through another path, e.g., Path 2 or even a shortest path discovered by the IP layer, to reach CE.

For node failure(s), the impact may become severe on the conventional multicast model and the cross-layer model, as the aggregated query and the multicast tree will be lost if there is no

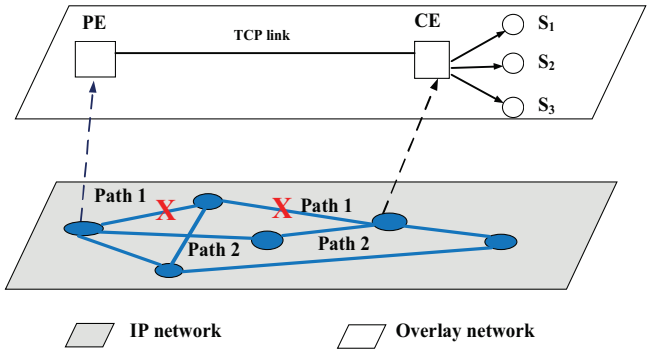

Fig. 4. An example scenario for network topology change.

special backup mechanism for intermediate brokers. We assume that failures will not happen for edge brokers, PE and CEs, as they are well protected or backed up in practice, and they are needed in all three system models. Under this assumption, intermediate node failure(s) have no effect on the peer model, as there is no need to recovery the aggregated subscriptions or the multicast tree for intermediate brokers.

3.4. Comparison and summary of different models

All three models described in this section can satisfy the functional requirements for an XML pub/sub system: the messages sent by the publisher can be correctly routed to interested subscribers. They all share some similar functionalities, e.g., XML query aggregation and XML publication message filtering. However, non-functional requirements (NFRs) are often the selection criteria for choosing among multiple design alternatives [4,39]. Two top NFRs are performance and maintenance; both significantly influence the decisions in choosing a particular design alternative. On the other hand, cost is the dominant nontechnical NFR. In communications systems, those three NFRs, i.e., performance, maintenance, and cost, are crucial selection criteria for various technologies.

Some widely adopted technologies in communications systems, e.g., the peer-to-peer model in VPNs, CDNs, and SDN, are also closely related to these three NFRs, although those three models provide similar functional requirements as their counterparts. Specifically, the peer-to-peer model used in VPNs is much more scalable than the overlay model, due to its simple management and provisioning feature [36]; CDNs significantly improve the efficiency for content delivery, as data are moved closer to the edges/users using the model [55]; and SDN is highly cost effective and adaptable as a result of decoupling the network control and forwarding functions [53]. Variations in architectural models can make substantial differences in NFRs. The following paragraphs highlight a comparison of those three XML pub/sub models from those three NFR perspectives.

In comparison to the conventional multicast model, both the cross-layer model and the peer model can effectively reduce the number of filtering and aggregation operations and the amount of query storage at the intermediate brokers. Both filtering and aggregation operations are time consuming for complex and nested queries; for the conventional multicast model, these two operations are performed at each broker, which significantly influence the efficiency. Without the need for the filtering operation at each intermediate broker for the cross-layer and the peer models, the E2E delay for message delivery to interested subscribers can be considerably lower.

In an extreme case where all intermediate brokers are also CEs, the cross-layer model becomes similar to the multicast model in that each intermediate broker (also a CE) uses specific paths based on the application-layer topology for message delivery and each broker also needs to perform the query aggregation and the filtering operations. But there are still differences among the three models for query aggregation:

- For the multicast model, each broker executes query aggregation for both queries directly received from subscribers and queries forwarded from the downstream brokers. One critical objective of the model is to obtain minimal aggregated query trees and related information; hence subscriptions have to be aggregated at each broker.
- For the cross-layer model, query aggregation is performed at each broker for the directly connected subscribers only, not for queries received from downstream brokers. As a result, the aggregated query tree at each broker for the cross-layer model is smaller than that used in the multicast model, which could reduce the query aggregation time. On the other hand, the amount of aggregated query information at a PE could be larger than that for the multicast model, but the additional amount is not significant because of large storage capacity of typical PEs.
- The peer model is similar to the cross-layer model, i.e., each broker performs the aggregation operation only for directly connected subscribers. But the peer model differs from the cross-layer model in the CEs transmitting the new queries to the PE via the network layer paths without having to go through the tree-like application-layer topology.

The filtering operation is also handled differently for these three models for the aforementioned extreme scenario.

- For the multicast model, as stated in Section 3.2, each broker performs filtering to identify both interested subscribers and matched downstream brokers. As queries are aggregated at each broker, the total number of messages that need to be delivered to all CEs is optimal [31]. However, a message needs to go through each broker along the path, even if some brokers may not have matched subscribers.
- For the cross-layer model, the PE identifies the matched CEs for each publication message and delivers the message to those CEs. Each broker then only identifies matched subscribers that are directly connected to itself without having to find matched brokers. The total number of messages that need to be delivered from a PE to CEs may be higher than that in the multicast model; however, the number of CEs (instead of subscribers) generally is not large in practice. For an extremely large-scale pub/sub systems, it would become similar to the CDN service, where CEs are distributed and placed closer to subscribers. On the other hand, the aggregated query tree at each broker or CE is smaller than that used in the multicast model, which reduces the filtering time.
- The peer model is similar to the cross-layer model in that the PE identifies the matched CEs and forwards the message to those CEs only without further filtering. But the message delivery from the PE to each matched CE is transmitted via shortest paths that are calculated with the network layer protocols instead of the tree-like application-layer topology that is used for the other two models. Even in the worst case, when all CEs are matched and a PE needs to transmit a message to all CEs for interested subscribers, the peer model can still improve efficiency (by using a fewer number of physical network devices in realistic networks compared to that used in the application layer, as demonstrated in Section 5.4) and reliability (in case a node fails as described in Section 3.3). In a WAN environment, the network propagation delay and the processing delay going through network devices are dominant factors in total E2E delay. Also, queries with containment relations can be stored at the query routing table of the PE multiple times and duplicated

messages may be sent over a common path to different CEs. But the number of CEs is typically not large in a pub/sub system.

Besides the two abovementioned key operations, the peer model does not need to create and maintain a multicast tree (and the concomitant state information for the multicast tree) compared to the other models. Building and maintaining a multicast tree is challenging from both the technical and nontechnical (e.g., cost) aspects, especially for large networks or multiple ISPs [58]. In fact, intermediate brokers are no longer needed for the peer model. As a result, the peer model is much more cost effective and flexible for path selection. The concept of moving intelligence and complexity to edge devices is similar to the practice used in existing communications systems to relieve the load in core network devices.

In addition, the peer model can handle network topology changes or failures efficiently compared to the cross-layer model and the multicast model without a need for a special backup mechanism for query and path recovery.

4. Experimental setup and performance metrics

This section describes how experiments are set up and the various parameters that are used. There are two sets of experiments. The first set of experiments is conducted in a controlled and isolated LAN and the link bandwidth is 100 Mbit/s. Each machine is a system consisting of two 3.0 GHz Intel Pentium cores with 4.0 GB memory running under Windows XP. The focus of the first set of experiments is on the impact of the filtering operation on the processing delay and the effect of various parameters in a controlled environment. Processing delay is the dominating factor in total end to end delay in a LAN environment, as the network propagation delay is negligible in a LAN. Note that due to resource limitation in our lab, multiple brokers are running on the same PC, (2–3 brokers for topology #1 as depicted in Fig. 5(a) and up to 5 brokers in topology #2 as shown in Fig. 5(b)). The performance trends, however, are similar to the case where a broker is running on a separate PC in a test case scenario.

The second set of experiments, as illustrated in Fig. 5(c), is performed using Amazon Elastic Compute Cloud (EC2). XML brokers are deployed over Amazon datacenters around the world to emulate the WAN. In a WAN environment, network delay is much longer compared to that in a LAN, which is realistic for the actual pub/sub services. The configurations of the EC2 instances used are t2.micro (1 vCPU, 1GB memory, Microsoft Windows Server 2012 R2 Base) [5].

4.1. Experimental network topologies

Three network topologies, as depicted in Fig. 5, have been used for the experiments. We choose a tree, as shown in Fig. 5(a), as topology #1 for the following reasons: (i) the topology is a representative balanced binary tree and there are shared paths from PE (broker A) to the leaf nodes (e.g., path A–B–C for brokers F and G); (ii) it was the left branch of an existing network topology tree used in a similar area [24].

Next, topology #2, a longer delivery path, is adopted (see Fig. 5(b)). The length of topology #2 is 11, which is considered long for a multicast tree path. The purpose of topology #2 is to examine the impact on the performance, particularly processing delay, of different models for XML routing when the length of an E2E delivery path is increased. A multicast path with depth of 11 indicates that there could be a large number of brokers if the multicast tree is close to a balanced tree (e.g., could have up to $\sim 2^{11}$ brokers).

Topology #3 is evaluated using the Amazon EC2 Web service. Amazon offers about a dozen datacenters across the world. Topology #3, as demonstrated in Fig. 5(c), depicts the eight XML brokers

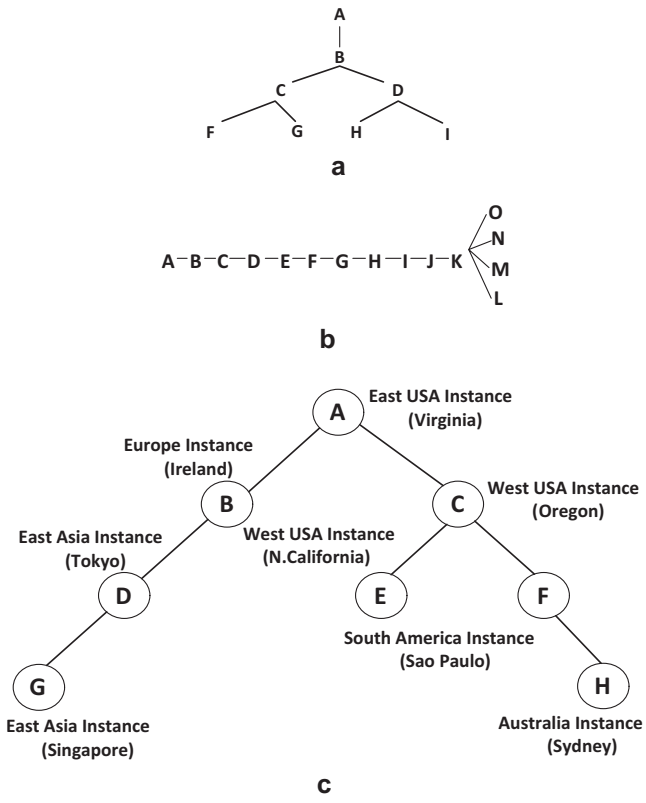

Fig. 5. Experimental network topologies: (a) topology #1; (b) topology #2; (c) topology #3 using Amazon EC2 Web service.

that have been used for our experiment, including the actual geographical location of each broker. In this experiment, broker A is the PE and all the other nodes (i.e., B–H) are CEs, i.e., each broker, except node A, is also a CE, which is the extreme scenario described in Section 3.4. The main purpose of this experiment is to investigate the effect of network propagation delay in a realistic WAN environment on overall performance of the different models. In practice, an XML pub/sub system may be geographically distributed, e.g., brokers could be far away from each other, and many physical network devices may be involved even though the application layer topology is straightforward. Therefore, the results of the experiment can reveal the real delay in a large geographically distributed network, which is often missing in the literature. Moreover, we also investigate the difference between the application-layer and the network layer topologies in terms of the number of the actual path lengths, which will be presented in Section 5.4.

4.2. Experimental parameters

Table 1 lists the parameters and their values used in experiments for different XML routing models. One parameter is varied in each experiment while the other parameters are held at their default values. Table 2 shows the default parameter values for XML messages and XPath queries. The exponential random numbers for message inter-arrival time are generated from SimJava [63].

Determination of default values for some experimental parameters. Default parameter values used in our experiments are based on important papers in the literature. For example, in [26], the default values for the experimental parameters are: query depth = 6, prob(//) = 0.2, prob(∗) = 0.1, number of predicates = 0, number of branches = 0 and query population = 20,000; in [24], the parameter values are listed as follows: query depth is

Table 1
Parameter values used in the experiments.

Parameters	Descriptions	Values
Message type	Test messages type	Recursive messages with book.dtd
Message size	Maximum test message size (KB)	{1, 5, 10, 15, 20}
Λ	Publication message arrival rate (msg/s)	{0.5, 0.85, 2, 4,8}
Number of XPath queries	Test queries population	{10,000, 20,000, 50,000}
Prob (//)	Probability that an element has //	{0.1, 0.2, 0.3}
Prob (∗)	Probability that an element has ∗	{0.1, 0.2, 0.3}
Prob (branching)	Probability that a query has branches	{0, 0.1, 0.2}
Prob(⌈⌉)	Probability that a query has predicates	{0.1, 0.2, 0.3}
Query depth	The longest path length of a test query tree	{3, 6, 10}
Match ratio	The ratio of matched queries over test query population	around {0.05}
Number of CEs	The number of CEs of the test network topology	{2, 4, 6}

Table 2
Default parameter values of XML messages and XPath queries.

Parameter	Value
Message size	10 KB
Λ	0.85 msg/s
XPath query population	20,000
Query depth	6
Prob (//)	0.2
Prob (∗)	0.2
Prob (branching)	0.1
Prob (⌈⌉)	0.2
Query duplication	Allowed
Match ratio	0.05
Number of CEs	4

between 3 and 10, prob(//) = 0.1, prob(∗) = 0.1, message recursion depth = 3, message arrival is characterized by a Poisson distribution with the arrival rate of 1 msg/s, query population is between 1000 and 50,000, and message size is 22 tag pairs; in [30], the parameter values used are described as follows: query population is between 1 and 50,000, query depth is between 6 and 10, prob(//) = 0.2, and prob(//) = 0.2. The default values for some of the system parameters are based on values used in the research described in the referred articles [26,30,24]. Note that the average arrival rate refers to the publication message arrival rate sent by a publisher, not the arrival rate for regular messages transmitted over the network. An arrival rate of 1 msg/s, for instance, is considered high for regular pub/sub systems. The number of query population, message size, the complexity of messages, e.g. recursive messages, and the query depth can significantly affect the performance. See Section 5.1 for some performance results related to those parameters.

We use the method of long run to capture the steady-state of the system. As time becomes longer, the effect of the initial conditions on later observations lessens and the observations appear to vary around a common mean. The data-collection phase can begin after this point is reached [7]. We analyze the length of experiments and measure the average E2E delay as a function of the experimental run length that is given by the number of messages considered in the experiment. The experimental run lengths used include 100, 250, 500, 1000, 2000, 3000, 5000, 6000, 7000, 10,000, and 12,000. From the absolute time difference between two consecutive run lengths in the list, we find that after a length of 5000, the absolute difference is smaller than 1.5% and the system reaches a steady-state. Hence, 6000 messages are selected as the run length to be used in the experiments. This was found to be adequate for the investigation of the relative performance of the algorithms that this paper focuses on. All reported data include outliers in the following experiments.

4.3. Performance metrics

The performance metrics used in the experiments are explained below.

- **Average E2E delay for one message.** This metric is calculated based on the following definition.

Definition 2. When the conventional multicast approach, cross-layer or the peer model is used, let t_{is} represent the timestamp for sending the ith publication message from the PE. Let t_{ij} represent the timestamp of the jth ACK message of the ith publication message received by the PE. Let d_{ij} be the difference in time between t_{ij} and t_{is}, $d_{ij} = t_{ij} - t_{is}$. Let k represent the total number of ACK messages corresponding to the same ith publication message received by the PE. Then, the average E2E delay for publication message i is $1/K \sum_{j=1}^{K} dij$ Let R be the number of messages. The average E2E delay for one message $= 1/R \sum_{i=1}^{R} (1/k \sum_{j=1}^{k} dij)$.

- **Average filtering time at one node.** The average filtering time is the time difference between the arrival time of a publication message at the filtering engine and the time the same message leaves the filtering engine. The average is computed over 100 runs.
- **Average round-trip time between two nodes.** The average round-trip time between two nodes is the time difference between the time an ACK message is received by a sender and the time of sending the publication message to the receiver by the sender. The average is computed over 200 runs.
- **Average transmission time at one node.** The average transmission time is the time difference between the time a publication message is starting to be sent and the time when sending is complete. The average is computed over 200 runs.
- **Average processing time at a broker.** The average processing time is the time difference between the time at which the message is received by a broker and the time the same message is sent by the broker. The average is computed over the total number of messages received by the broker.

5. Performance results and analysis

In this section, we present three sets of experiments in order to quantify the performance of the various approaches (multicast, cross-layer, and peer models). Specifically, four sets of experimental results are presented in the following sub-sections. The first set of experiments is described in Section 5.1, which is performed to measure and better understand the complex XML filtering operations. Sections 5.2–5.4 present performance results for topology #1–topology #3, respectively.

Table 3
Performance measurements between two nodes for different messages: (a) average transmission time; (b) average round-trip time.

(a)		(b)	
Message size (KB)	Average transmission time (ms)	Message size (KB)	Average round-trip time (ms)
1	0.13	1	0.65
5	0.2	5	1.07
10	0.26	10	1.12
15	0.36	15	1.29
20	0.44	20	1.46

Table 4
Average filtering time for one message: (a) the message size is varied; (b) only one query parameter is varied; (c) the query depth is varied.

(a)

Message size (KB)	1	5	10	15	20
Average filtering time for 1 message (ms)	11.66	64.69	116.33	160.83	245.18

(b)

Varying query parameter	Average filtering time for 1 message (ms)
Default	116.33
Prob (//) = 0.1	77.13
Prob (//) = 0.3	141.55
Prob (∗) = 0.1	99.85
Prob (∗) = 0.3	115.14
Prob (branching) = 0	79.11
Prob (branching) = 0.2	128.06
Prob ([]) = 0.1	110.98
Prob ([]) = 0.3	116.34
Query population = 10,000	75.57
Query population = 50,000	170.39

(c)

Varying longest query depth	Average filtering time for 1 message (ms)
Query depth = 3	60.59
Query depth = 6	375.53
Query depth = 10	1351.84

5.1. Experimental results for primary operations of XML filtering and matching conducted at one broker

The performance of primary XML filtering operations conducted at one broker is first examined. The purpose of the results presented in this section is to provide a reference to the results in other sections.

Table 3(a) records the average transmission time for publication messages of different sizes. Table 3(b) records the average round trip time between two nodes for simply sending a message to a client. As discussed in Section 4.2, the results presented in Table 3 are the average of 6000 runs. The network delay for our prototype is negligible in a LAN, which is consistent with the expectation. The average message transmission time for real core routers is even smaller due to the higher link bandwidth, e.g., 10–100 Gbps, they have. Compared to the filtering cost, the message transmission time are negligible. Further, the delay caused by the network is the primary cause of the difference between the performance of the cross-layer model and the peer model. Consequently, the performance difference between these two models is small in a LAN environment, which is consistent based on our observation. Hence, in Sections 5.2 and 5.3, only the comparison of the traditional XML multicast model and the cross-layer model is presented for the LAN environment. In addition, the variations in different experimental results are statistically insignificant in a LAN environment for topology #1 and topology #2. For topology #3, the 95% confidence levels for various experiments are calculated and presented in Section 5.4.

Table 4(a) shows the performance of Yfilter when message size is varied and other parameters are at default values as shown in Table 2. Table 4(b) lists the filtering performance when only one parameter of an XPath query is changed. Table 4(c) shows the filtering performance when query depth is varied. The purpose is to study the impact of different parameters on the overall performance.

The *book.dtd* is used for the experiments and the DTD has one long branch and some short branches. To generate a query with long depth, we modify the code of *PathGenerator* class in Yfilter by allowing the long branch containing a recursive element *section* to be the main path. But for queries in previous experiments, every branch has the same probability of being the main path. The test message *book.xml* is a complex recursive XML message. The theoretical analysis in [10] indicates that the number of matched transitions from a node is an exponential function of the recursive depth of a message and queries. The results of Table 4(c) show that the filtering cost increases substantially for a recursive message as the query recursion depth increases.

5.2. Experimental results for topology #1

This section presents the experimental results for topology #1, as depicted in Fig. 5(a). Table 5 shows the performance results using the default parameter values with the two approaches: conventional filter-based XML multicast model and cross-layer model. (The difference between the cross-layer model and the peer model is small in a LAN.) The average E2E delay for one message using

Table 5
Performance evaluation for default parameter values with topology #1.

Performance metrics	Multicast model	Cross-layer model
Average E2E delay for one message (ms)	316.35	63.47

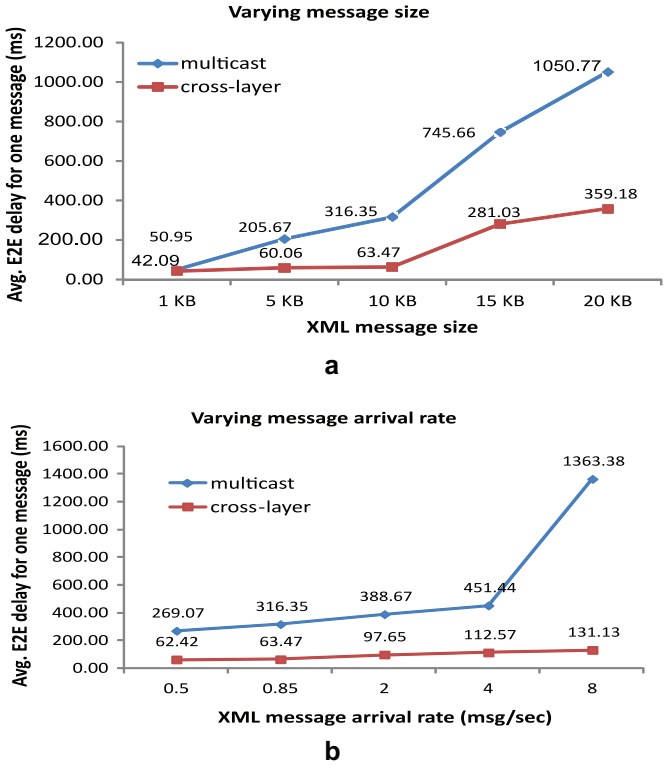

Fig. 6. Performance evaluation for topology #1: (a) varying XML message size; (b) varying default message arrival rate.

multicast is 4.98 times that of the cross-layer model. As can be seen, the cross-layer model notably outperforms the multicast approach.

Table 6 lists the average processing time at each broker for topology #1. Node A is the PE and it only sends messages to the next broker, B. From Table 6, we can find a big difference in performance using different communication models for intermediate brokers. For example, the average processing times for brokers B, C, and D when the multicast model is used are 108.4, 120., and 109.2 ms, respectively, while the average processing time when the cross-layer model is applied are only 2.4, 5.2, and 5.3 ms, respectively.

The purpose of the analysis presented in Fig. 6 is to understand the effect of message parameters on the multicast model and the cross-layer model.

Effect of message size. Fig. 6(a) shows the performance of E2E delay as a function of the message size. The cross-layer model outperforms the multicast model. For example, the average E2E delays are 50.95 and 1050.77 ms using conventional filter-based XML multicast model for a 1 and a 20 KB message, respectively. This shows the limitations of the conventional filter-based XML multicast model: expensive filtering operations have to be repeated many times.

Effect of message arrival rate. Fig. 6(b) shows the performance of the average E2E delay as a function of the average message arrival rate. The message with default message size and the queries generated from the default query parameters in Table 2 are used in the experiments; but the message arrival rate is varied. For multicast, the average E2E delay increases much faster than that of the cross-layer model (see Fig. 6(b)).

Effect of query parameters. The purpose of the analysis shown in Fig. 7 is to investigate the effect of query parameters on the performance of the multicast model and the cross-layer model. In Fig. 7, we vary the parameters of an XPath query, including the probability of //-operator, ∗-operator, predicates, branching, query population, and query path. The results from Fig. 7(a)–(d) demonstrate that the cross-layer model always outperforms the conventional filter-based XML multicast model. Fig. 7(d) depicts the effect of the depth of recursive queries on the filtering performance. Complex queries have an obvious impact on the performance of the conventional multicast model, as the processing time will increase considerably. In comparison to the cross-layer model, the average E2E delay for the conventional filter-based XML multicast model increases much more sharply with an increase in the XPath query depth.

5.3. Experimental results for topology #2

This section presents the experimental results for topology#2, as depicted in Fig. 5(b). The purpose of studying topology #2 is to investigate the effect of long delivery paths for a potentially large multicast tree on the performance of the different approaches.

Table 7 shows the performance results for default parameter values using these two models. The average E2E delay for the conventional filter-based XML multicast model is 10.7 times that of the cross-layer model in terms of the average E2E delay. Table 8 lists the average processing time at each broker for topology #2. The experimental results shown in both tables demonstrate that the cross-layer model approach outperforms the conventional filter-based XML multicast model.

The purpose of the analysis presented in Fig. 8 is to investigate the effect of message parameters on the performance of the conventional filter-based XML multicast model and the cross-layer model.

Effect of message size. Fig. 8(a) shows the results when the message size is varied. We observe that the message size can

Table 7
Performance evaluation for default parameter values with topology #2.

Performance metrics	Multicast model	Cross-layer model
Average E2E delay for one message (ms)	1270.26	118.63

Table 6
Average processing time using default parameter values for topology #1.

Average processing time for one message (ms)	B	C	D	F	G	H	I
Multicast	108.44	120.20	109.18	53.06	55.52	43.43	50.53
Cross-layer	2.42	5.19	5.33	55.05	52.25	43.64	49.41

Table 8
Average processing time using default parameter values for topology #2.

Brokers	B	C	D	E	F	G	H
Multicast	122.54	119.06	120.54	123.41	115.69	117.08	117.97
Cross-layer	1.67	1.88	1.83	1.63	1.43	1.74	1.75
Brokers	I	J	K	L	M	N	O
Multicast	117.44	117.9	116.53	92.58	94.35	64.92	73.77
Cross-layer	1.83	1.70	22.77	95.75	75.31	66.17	86.19

Table 9
Comparison of average E2E delay for three models on Amazon Cloud.

System workload	Conventional multicast	Cross-layer	Peer model
Only one query	302.68 ms	285.77 ms	120.81 ms
Default workload (20,000 queries)	253.06 ms	238.74 ms	90.89 ms

significantly affect the performance of multicast because the filtering cost increases as the message size increases, and the filtering operation is executed at each broker. The average E2E delay for the conventional filter-based XML multicast model includes the filtering time and transmission time at all brokers along the path from PE to CEs. The average E2E delay for the cross-layer model includes the times for computing the next hop and the new destination list as well as the times for filtering at CEs.

Effect of message arrival rate. Fig. 8(b) shows the trend when the message arrival rate increases. If λ is smaller than 90% usage of the maximum processing ability, e.g., 0.5, 0.85, and 2 msg/s, the change on the average E2E delay is small. However, for the multicast model, when λ reaches a certain number, e.g., 4 msg/s, the average E2E delay increases quickly because of the queue waiting time. The performance improvement produced by the cross-layer model increases with an increase in λ.

Effect of query parameters. The purpose of the analysis shown in Fig. 9 is to investigate the effect of critical query parameters on the performance of multicast and cross-layer model approaches. We experiment with different values for query parameters, including prob(//), prob(∗), prob[], prob(branching), and query population. The message and the λ are based on default values.

First, we can observe that the cross-layer model outperforms the XML multicast model in different query parameter combinations. Second, the filtering cost increases as the complexity of XPath queries (captured in prob(//) in Fig. 9(a), prob(branching) in Fig. 9(b), and query population in Fig. 9(c)) increases. But the performance for various probability values for predicates, prob[], as presented in Fig. 9(a), shows minor changes due to the fact that more query predicates do not match the message. Similar to the results for topology #1, Fig. 9(d) demonstrates that query depth has a considerable effect on processing time due to the complex recursive structure. More efficient algorithms can be studied for complex queries.

5.4. Experimental results for topology #3

This section presents the experimental results for topology #3, as depicted in Fig. 5(c), in a cloud environment. Each broker is running on a separate Amazon datacenter and brokers are far away from each other. The main purpose of this experiment is to investigate the effect of propagation time and network overheads on E2E delay in a realistic WAN environment.

Table 9 demonstrates the average E2E delay results for two test cases; each test case contains three XML pub/sub models, e.g., multicast, cross-layer, and the peer models. The same workload has been adopted for all experiments. Measurements for each test case (including the three models) have been collected during the same period of time to reduce the impact of different traffic patterns due to different time zones during a day. But those two different test cases, as shown in Table 9, were conducted at different time periods of the day, hence the delay caused by the Internet may have different impact on those two test cases. In this experiment, broker A acts as the PE and the rest of the brokers are CEs. Each measurement is the average of 6000 runs.

We have also performed a number of additional experiments in a similar cloud environment using all five message sizes (1, 5, 10, 15, and 20 KB) and all five arrival rates (0.5, 0.85, 2, 4, and 8 msg/s) as listed in Table 1 and the default values for various other parameters that are shown in Table 2. Further, for each configuration, we also considered two different scenarios: a high traffic scenario for North America (typically experienced during business hours) and a low traffic scenario for North America (typically experienced during non-business hours). Note that the data centers used for our experiments are spread over the world; hence, the time zones and traffic volumes vary for those locations. Each experiment was carried out for 1000 runs. In other words, each experiment with the same configuration was repeated during two different times of the day and the results of 1000 runs were collected for each experiment.

We calculated 95% confidence intervals for those experiments. The margin of error (i.e., the range of values above and below the sample mean) for different models is in the range between $\pm$1.3% and $\pm$2.3% for various message sizes while the default values are used for other parameters (shown in Table 2). However, the margin of error for different arrival rates varies more when other parameters are fixed with the default values. For the peer model, the range for margin of error is the smallest from $\pm$1.4% to $\pm$3.9% for our experiments; for the conventional model, the range is the largest from $\pm$1.8% (for 0.5 msg/s) to $\pm$12.2% (for 2 msg/s), but most are under 8% (e.g., $\pm$1.9%, $\pm$4.3%, $\pm$6.0%, $\pm$7.3% and $\pm$7.9%). Further, the resulting intervals do not overlap: the E2E delay results even in the worst case (the highest values above the sample means) for the peer model are still better than those of the best case for the conventional model (the lowest values below the sample means).

Note that for the multicast model and the cross-layer model, some CEs are *logically* closer to PE (broker A) at the application layer, e.g., brokers B and C are only logically one-hop away from broker A, whereas G and H are three-hop away. However, for the peer model, PE is only one-hop away from all CEs logically at the application layer. The forwarding paths from PE (node A) to CEs for the peer model are determined using the network layer routing technologies (e.g., OSPF and BGP protocols) instead of the multicast tree. The results shown in the table are the average E2E delay for all brokers.

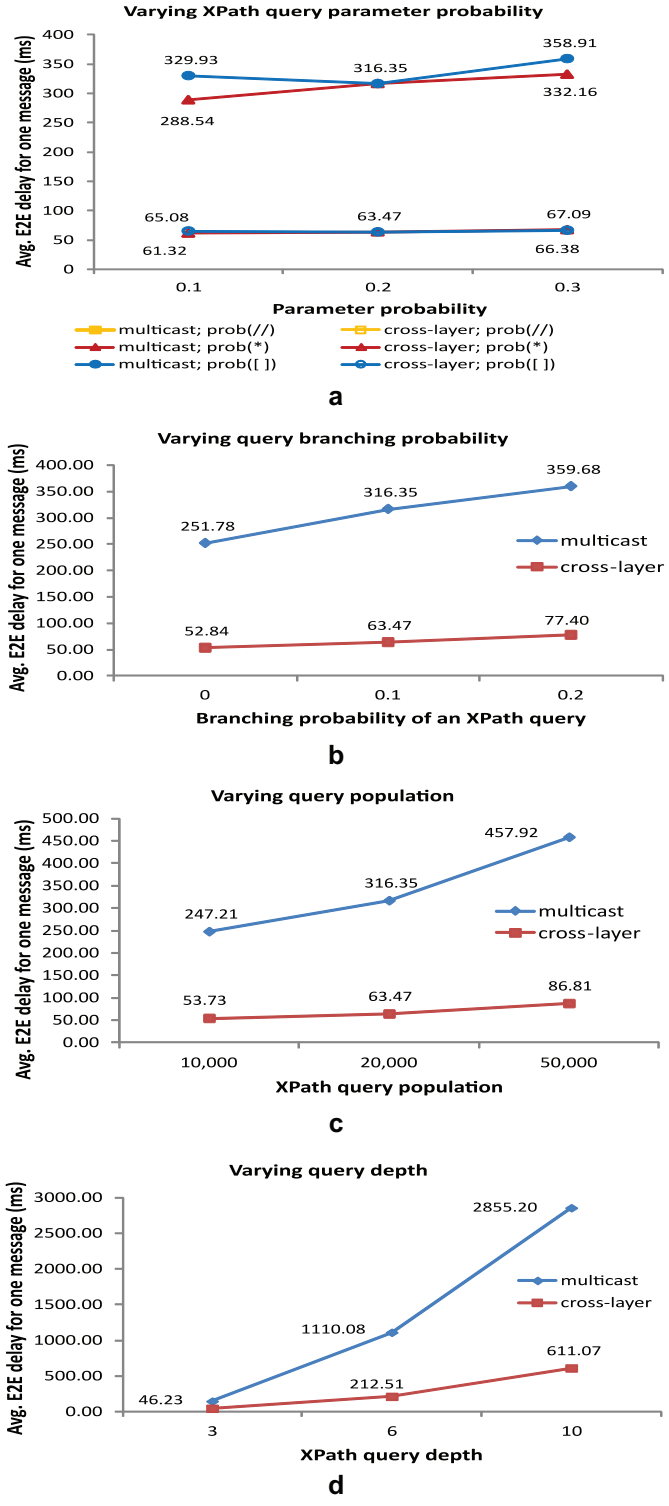

Fig. 7. Performance evaluation for topology #1: (a) varying prob (//), prob (∗), and prob ([ ]); (b) varying XPath query branching probability; (c) varying XPath query population; and (d) varying XPath query depth.

As shown in Table 9, the conventional multicast approach still has the highest delay for both test cases. The peer model reduces the E2E delay by more than 60% in comparison to that of the multicast model. However, the gap between the cross-layer model and the multicast model has significantly reduced compared to the results obtained from a LAN environment where the processing time is the dominating factor. The reason is that the messages traverse

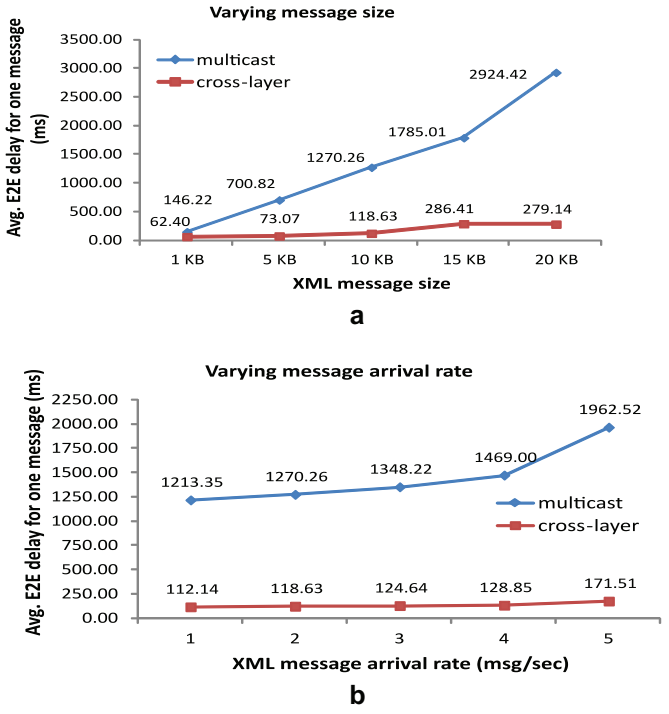

Fig. 8. Performance evaluation for topology #2: (a) varying XML message size; (b) varying default message arrival rate.

Table 10
Number of hops for cross-layer model and peer model using *traceroute*.

Path	Multicast/Cross-layer	Peer model
A -> G	36	16
A -> E	25	12
A -> H	50	13

through the same path from the PE to CEs based on the multicast tree (as configured in Fig. 5(c)) for both multicast and cross-layer models, and the propagation delay is the primary contributing factor for a geographically large WAN.

On the other hand, in the peer model, for traffic between different datacenters, the routing paths typically only follow the public Internet routing or specific traffic engineering design, but there is no need to follow the path of the multicast tree. The actual paths to some CEs could be shorter than that used in the multicast tree. Table 10 confirms that the paths are shorter from PE (broker A) to CEs in the peer model by displaying the number of hops (routers or switches) obtained from using the *traceroute* command. The results in Table 10 are the number of physical hops for the multicast/cross-layer and the peer models from the PE to CEs (G, E, and H), respectively, although the logical application-layer topology may be similar. As can be seen from this table, the number of physical hops is considerably larger for the multicast and cross-layer models in comparison to the peer model. As a result, the E2E delay for the cross-layer model is much higher than that of the peer model and the gap between the multicast model and the cross-layer model also has reduced considerably due to the long paths via a large number of common hops from PE to CEs for these two models.

6. Conclusions and future directions

XML pub/sub systems are important applications and the performance of such systems is important as the data volume keeps

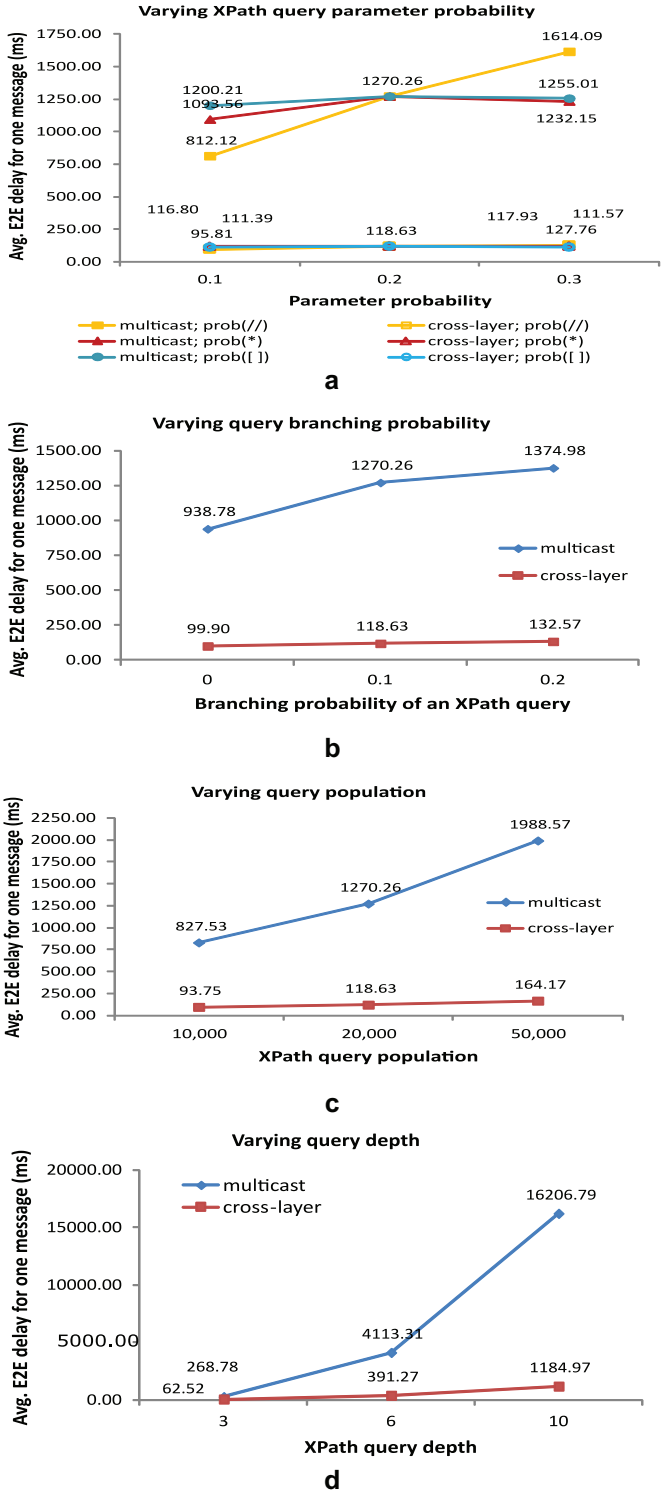

Fig. 9. Performance evaluation for topology #2: (a) varying the probability of prob (//), prob (∗), and prob ([]); (b) varying XPath query branching probability; (c) varying XPath query population; and (d)varying XPath query depth.

increasing. Performance improvement can be realized with various techniques. This paper focused on performance improvement from the communication models perspective. Effective communication models have potential to significantly improve the performance, as stated in Section 1. This paper presented two communication models for XML pub/sub services, i.e., the cross-layer model and the peer model. We then compared these two models with the conventional filter-based XML multicast model. The conventional XML multicast model builds an application-layer broker overlay above the network layer. The approach forwards the minimal number of XML messages to subscribers at the expense of two high computational cost of XML query aggregation and message filtering operations performed repeatedly at each hop from a PE broker to a CE broker.

Both the proposed cross-layer model and the peer model limit the two expensive XML broker operations to be performed only at PE and CE nodes. However, the cross-layer model still relies on XML-capable brokers for message forwarding. On the other hand, the peer model does not establish and maintain a multicast tree, and message forwarding between PE and CEs is realized using the regular network routing technologies rather than following the multicast tree paths. The peer model offers several advantages: low computational cost (both query aggregation and publication filtering operations), easy deployment and management without the need of setting up application-layer XML-capable brokers, high network efficiency, support for frequent user subscription changes, and resilient of broker failures (assuming no failures occur at PE and CEs).

An XML pub/sub prototype was devised and the different routing models were evaluated in a LAN environment and on the Amazon cloud centers around the world to emulate a WAN environment where propagation delay becomes the dominant factor. We compared three different models through extensive experimentation. The results showed that the cross-layer model and the peer model significantly outperformed the multicast model in a LAN environment primarily due to the elimination of the expensive filtering operation performed on each broker. Further, the peer model has a notably lower E2E delay than the other two models in a WAN environment as a result of eliminating the filtering operation performed at intermediate brokers and shorter paths used at the network layer than that of the application layer.

The focus of this paper was on publication message forwarding. Integrating the publication message routing and user query aggregation techniques into a prototype and evaluating its performance with real workload warrants investigation.

Acknowledgement

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Ontario Centres of Excellence (OCE).

References

[1] M. Abdul Nizar, G.S. Babu, P.S. Kumar, SFilter: A simple and scalable filter for XML streams, in: Proceedings of the International Conference on Management of Data, Citeseer, 2009.
[2] M. Alrammal, G. Hains, A research survey on large XML data: Streaming, selectivity estimation and parallelism, Inter-Cooperative Collective Intelligence: Techniques and Applications, Springer, 2013, pp. 167–202.
[3] M. Altinel, M. Franklin, Efficient filtering of XML documents for selective dissemination of information, in: Proceedings of the 26th International Conference on Very Large Data Bases (VLDB), ACM VLDB Endowment, 2000, pp. 53–64.
[4] D. Ameller, C. Ayala, J. Cabot, X. Franch, Non-functional requirements in architectural decision making, IEEE Softw. 30 (2) (2013) 61–67.
[5] Amazon EC2 Web Service. http://aws.amazon.com/ec2, 2015 (accessed February 2015).
[6] R. Baldoni, R. Beraldi, L. Querzoni, A. Virgillito, Efficient publish/subscribe through a self-organizing broker overlay and its application to siena, Comput. J. 50 (2007) 444–459.
[7] J. Banks II, J.S. C., B.L. Nelson, D.M. Nicol, Discrete-Event System Simulation, third ed, Prentice Hall, 2000.
[8] B. Bou-Diab (2005). Virtual private network publish-subscribe multicast service. US patent, patent No: US7797382B2 http://www.freepatentsonline.com/7797382.html (accessed May 2015).
[9] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, F. Yergeau. (2009). Extensible markup language (xml). http://www.w3.org/TR/2006/REC-xml11-20060816/ (accessed May 2015).

[10] K.S. Candan, W.-P. Hsiung, S. Chen, J. Tatemura, D. Agrawal, Afilter: Adaptable xml filtering with prefix-caching suffix-clustering, in: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), ACM VLDB Endowment, 2006, pp. 559–570.
[11] F. Cao, J.P. Singh, Medym, Match-early and dynamic multicast for content-based publish-subscribe service networks, in: Proceedings of 4th Intl. Workshop on Distributed Event-Based Systems (DEBS), ACM, 2005, pp. 370–376.
[12] F. Cao, J.P. Singh, Medym: Match-early with dynamic multicast for content-based publish-subscribe networks, in: Proceedings of the ACM/IFIP/USENIX International Conference on Middleware (Middleware), Springer-Verlag, 2005, pp. 292–313.
[13] Y. Cao, C.-H. Lung, S. Majumdar, A subscription coverage technique for xml message dissemination, in: Proceedings of the 9th IEEE Annual Intl. Symp. on Applications and the Internet (SAINT), 2009, pp. 137–140.
[14] Y. Cao, C.-H. Lung, S. Majumdar, A peer-to-peer model for xml publish/subscribe services, in: Proceedings of the 9th Annual Communication Networks and Services Research Conference (CNSR), 2011, pp. 26–32.
[15] Y. Cao, C.-H. Lung, S. Majumdar, An XPath query aggregation algorithm using a region encoding, in: Proceedings of IEEE/IPSJ 11th Annual Intl. Symp. on Applications and the Internet (SAINT), 2011, pp. 27–36.
[16] A. Carzaniga, D.S. Rosenblum, A.L. Wolf, Design and evaluation of a wide-area event notification service, ACM Trans. Comput. Syst. 19 (2001) 332–383.
[17] A. Carzaniga, M.J. Rutherford, A.L. Wolf, A routing scheme for content-based networking, in: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), 2004, pp. 918–928.
[18] A. Carzaniga, A.L. Wolf, Forwarding in a content-based network, in: Proceedings of the ACM Conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM), 2003, pp. 163–174.
[19] M. Castro, P. Druschel, A.-M. Kermarrec, A. Rowstron, Scribe: A large-scale and decentralized application-level multicast infrastructure, IEEE J. Select. Areas Commun. 20 (2002) 1489–1499.
[20] C.Y. Chan, P. Felber, M. Garofalakis, R. Rastogi, Efficient filtering of xml documents with xpath expressions, VLDB J. 11 (2002) 354–379.
[21] R. Chand, Large dissemination of information in publish-subscribe systems Ph.D. thesis, University of Nice Sophia-Antipolis, 2005.
[22] R. Chand, P.A. Felber, A scalable protocol for content-based routing in overlay networks, in: Proceedings of 2nd IEEE International Symposium on Network Computing and Applications (NCA), 2003, pp. 123–130.
[23] R. Chand, P. Felber, Xnet: A reliable content-based publish/subscribe system, in: Proceedings of 23rd IEEE International Symposium on Reliable Distributed Systems (SRDRS), IEEE, 2004, pp. 264–273.
[24] R. Chand, P. Felber, Scalable distribution of xml content with xnet, IEEE Trans. Parallel Distrib. Syst. 19 (2008) 447–461.
[25] C. Chen, H.A. Jacobsen, R. Vitenberg, Divide and Conquer Algorithms for Publish/Subscribe Overlay Design, in: Proceedings of IEEE 30th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2010, pp. 622–633.
[26] S. Chen, H.-G. Li, J. Tatemura, W.-P. Hsiung, D. Agrawal, K.S. Candan, Scalable filtering of multiple generalized-tree-pattern queries over xml streams, IEEE Trans. Knowl. Data Eng. 20 (2008) 1627–1640.
[27] G. Chockler, R. Melamed, Y. Tock, R. Vitenberg, Constructing scalable overlays for pub-sub with many topics, in: Proceedings of the 26th annual ACM symposium on Principles of distributed computing (PODC), 2007, pp. 109–118.
[28] J. Clark, S. DeRose. (1999). XML path language (XPath) 1.0. http://www.w3.org/TR/xpath/ (accessed May 2015).
[29] L. Dai, C.-H. Lung, S. Majumdar, Bfilter- efficient XML message filtering and matching in publish/subscribe systems, J. Softw. 11 (4) (2016) 376–402.
[30] Y. Diao, M. Altinel, M.J. Franklin, H. Zhang, P.M. Fischer, Path sharing and predicate evaluation for high-performance xml filtering, ACM Trans. Database Syst. 28 (2003) 467–516.
[31] Y. Diao, M.J. Franklin, Query processing for high-volume xml message brokering, in: Proceedings of the 29th International Conference on Very Large Data Bases (VLDB), ACM VLDB Endowment, 2003, pp. 261–272.
[32] Y. Diao, S. Rizvi, M.J. Franklin, Towards an Internet-scale xml dissemination service, in: Proceedings of the 30th International Conference on Very Large Data Bases (VLDB), ACM VLDB Endowment, 2004, pp. 612–623.
[33] E.-S.M. El-Alfy, S. Mohammed, A.F. Barradah, XHQE: A hybrid system for scalable selectivity estimation of XML queries, Inform. Syst. Front. (June 2015) 1–17.
[34] W. Fenner, D. Srivastava, Xtreenet: Scalable overlay networks for xml content dissemination and querying, in: Proceedings of the 10th IEEE International Workshop on Web Content Caching and Distribution (WCW), 2005, pp. 41–46.
[35] M. Fu, Y. Zhang, Homomorphism resolving of xpath trees based on automata, in: Proceedings of a joint conference of the 9th Asia-Pacific Web Conference and the 8th International Conference on Web-Age Information Management (APWeb/WAIM), 4505, Lecture Notes in Computer Science, 2007, pp. 821–828.
[36] L.D. Ghein, MPLS Fundamentals, Cisco Press, 2006.
[37] M. Hofmann, L.R. Beaumont, Content Networking: Architecture, Protocols, and Practice, Morgan Kaufmann Publisher, 2005.
[38] W.-C. Hsu, C.-F. Li, I.-E. Liao, EFilter: An efficient filter for supporting twig query patterns in XML streams, in: Proceedings of International Conference on e-Business, SciTePress, 2013, pp. 1–8.
[39] ISO/IEC/IEEE 15288. (2015). System Requirements. International Organization for Standardization / International Electrotechnical Commissions/Institute of Electrical and Electronics Engineers.
[40] W.-C. Hsu, I.-E. Liao, CIS-X: A compacted index scheme for efficient query evaluation of XML documents, Inform. Sci. 241 (2013) 195–211.
[41] J. Kwon, P. Rao, B. Moon, S. Lee, FiST: Scalable XML document filtering by sequencing twig patterns, in: Proceedings of 31st International Conference on Very Large Databases (VLDB), ACM VLDB Endowment, 2005, pp. 217–228.
[42] B. Lantz, B. Heller, K. McKeown, A network in a laptop: Rapid prototyping for software-defined networks, in: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, ACM, 2010, pp. 1–6.
[43] F. Li, H. Wang, L. Hao, J. Li, H. Gao, Approximate joins for XML at label level, Inform. Sci. 282 (2014) 237–249.
[44] G. Li, S. Hou, H.-A. Jacobsen, Routing of xml and xpath queries in data dissemination networks, in: Proceedings of the 28th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2008, pp. 627–638.
[45] J. Liu, Z.M. Ma, Y. Yan, Efficient labeling scheme for dynamic XML trees, Inform. Sci. 221 (2013) 338–354.
[46] X. Liu, L. Chen, C. Wan, D. Liu, X. Xiong, Exploiting structures in keyword queries for XML search, Inform. Sci. 240 (2014) 56–71.
[47] J. Liu, Z.M. Ma, X. Feng, Answering ordered tree pattern queries over fuzzy XML data, Knowl. Inform. Syst. 43 (2) (2015) 473–495.
[48] J. Lu, T.W. Ling, Z. Bao, C. Wang, Extended XML tree pattern matching: Theories and algorithms, IEEE Trans. Knowl. Data Eng. 23 (3) (2011) 1–15.
[49] V. Joseph, S. Mulugu. (2011). Developing Next Generation Multicast-Enabled Applications, Morgan Kaufmann.
[50] R. Martins, J. Pereira, WFilter: Efficient XML filtering for large scale publish/subscribe systems, in: Proceedings of Symposium on INForum, 2011, pp. 1–14.
[51] R. Moussalli, M. Salloum, W. Najjar, V.J. Tsotras, Massively parallel XML twig filtering using dynamic programming on FGPAs, in: Proceedings of IEEE 27th International Conference on Data Engineering (ICDE), 2011, pp. 948–959.
[52] R. Moussalli, R. Halstead, M. Salloum, W. Najjar, V.J. Tsotras, Efficient XML path filtering using GPUs, in: Proceedings of the 2nd International Workshop on Accelerating Data Management Systems using Modern Processor and Storage Architectures (ADMS), ACM, 2011, pp. 9–18.
[53] ONF : Open Networking Foundation (2015). *Software-Defined Networking (SDN) Definition*. https://www.opennetworking.org/sdn-resources/sdn-definition.
[54] O. Papaemmanouil, U. Cetintemel, Semcast: Semantic multicast for content-based data dissemination, in: Proceedings 21st IEEE International Conference on Data Engineering (ICDE), 2005, pp. 242–253.
[55] M. Pathan, R.K. Sitaraman, D. Robinson (Eds.), Advanced Content Delivery, Streaming, and Cloud Services, Wiley, 2014.
[56] P. Pietzuch, D. Eyers, S. Kounev, B. Shand, Towards a common API for publish/subscribe, in: Proceedings of the Inaugural International Conference on Distributed Event-based Systems (DEBS), ACM, 2007, pp. 152–157.
[57] P.R. Pietzuch, J. Bacon, Hermes: A distributed event-based middleware architecture, in: Proceedings 22nd International Conference on Distributed Computing Systems Workshops, IEEE, 2002, pp. 611–618.
[58] B. Quinn, K. Almeroth, IP multicast applications: Challenges and solutions, in: IETF RFC 3170, 2001 https://www.ietf.org/rfc/rfc3170.txt.
[59] P. Rao, J. Cappos, V. Khare, B. Moon, B. Zhang, Net-x: Unified data-centric internet services, in: Proceedings of the 3rd USENIX International Workshop on Networking Meets Databases (NetDB), 2007, pp. 1–6.
[60] SAX (2001). Sax: Simple api for xml. http://www.saxproject.org/ (accessed May 2015).
[61] P. Saxena, R. Kamal, System architecture and effect of depth of query on XML document filtering using PFilter, in: Proceedings of the 6th International Conference on Contemporary Computing (IC3), IEEE, 2013, pp. 192–195.
[62] R. Shi, F. Liu, Y. Zhang, B. Cheng, J. Chen, A mid-based load balancing approach for topic-based pub-sub overlay construction, Tsing. Sci. Technol. 16 (2011) 589–600.
[63] SimJava (2004). Simjava. http://www.dcs.ed.ac.uk/home/hase/simjava/ (accessed May 2015).
[64] W. Sun, Y. Qin, P. Yu, Z. Zhang, Z. He, HFilter: Hybrid finite automaton based stream filtering for deep and recursive XML data, Database Exp. Syst. Appl., Lect. Notes Comput. Sci. 5181 (2008) 566–580.
[65] A. Termehchy, M. Winslett, Using structural information in XML keyword search effectively, ACM Trans. Database Syst. 36 (1) (2011) 1–49.
[66] B. Williamson, Developing IP Multicast Networks, CISCO Press, 1999.
[67] X. Wu, D. Theodoratos, A survey on XML streaming evaluation techniques, Int. J. Very Large Data Bases 22 (2) (2013) 177–202.
[68] XPath 3.0. (2014). XML Path Language (XPath) 3.0. http://www.w3.org/TR/xpath-30/ (accessed May 2015).
[69] S. Yoo, J.H. Son, M.H. Kim, An efficient subscription routing algorithm for scalable xml-based publish/subscribe systems, J. Syst. Softw. 79 (2006) 1767–1781.
[70] H. Zhao, W. Xia, J. Zhao, The research on xml filtering model using lazy dfa, J. Softw. 7 (8) (2012) 1759–1766.